ADVANCED DISTRIBUTED SIMULATION TECHNOLOGY II (ADST II)

DISMOUNTED WARRIOR NETWORK ENHANCEMENTS FOR RESTRICTED TERRAIN (DWNERT)

DO #0055

CDRL AB02

DISAF MOUT Enhancements Final Report

For:

United States Army
Simulation, Training, and Instrumentation Command
12350 Research Parkway
Orlando, Florida 32826-3224
Attn: AMSTI-ET

By:

Science Applications International Corporation
12479 Research parkway
Orlando, FL 32826-3248

Lockheed Martin
Information Systems Company
12506 Lake Underhill Road
Orlando, FL 32825

Document Control Information

Revision History	Date
Initial release	9/30/1998

TABLE OF CONTENTS

EXECUTIVE SUMMARY **1**

1. INTRODUCTION **2**

1.1 PROJECT BACKGROUND 2
1.2 GENERAL REQUIREMENTS 2
1.3 STATEMENT OF WORK 3
1.4 DELIVERABLES 3
1.4.1 Hardware and Software *3*
1.4.2 CDRLs *3*

2. SOFTWARE DEVELOPMENT PROCESS **3**

2.1 OVERVIEW 4
2.2 NATURAL LANGUAGE CIS 5
2.3 SOFTWARE CIS 5
2.4 VERIFICATION AND VALIDATION 6
2.5 SOFTWARE CONFIGURATION MANAGEMENT 6
2.6 ON-LINE DOCUMENTATION 6
2.7 PROBLEM TRACKING REPORT DATABASE 7
2.8 SOFTWARE METRICS 7

3. TERRAIN DATABASE ENHANCEMENTS **7**

3.1 CREATING AN MES FOR THE MCKENNA MOUT TDB 7
3.1.1 Overview *7*
3.1.2 MES Plug-in Development *7*
3.1.3 Editting the MES .rdr File *8*
3.2 DYNAMIC MES MODIFICATION 9

4. NEW IC ENTITY DEFINITION **9**

4.1 NEW ENTITY TYPES 9
4.2 BODY COMPONENT DEFINITION 11
4.3 LOWER BODY 11
4.4 UPPER BODY 12
4.5 EYES 12
4.5.1 Requirements *12*
4.5.2 Existing ModSAF model *13*
4.5.3 DISAF modifications *13*
4.6 IC WEAPON 13
4.6.1 IC Weapon capabilities *13*
4.6.2 Weapon usage *14*
4.7 PHYSICAL DIMENSIONS 14
4.7.1 Dynamic dimensions *14*
4.7.2 Dynamic sensor location *15*

5. INDIVIDUAL BEHAVIOR **15**

5.1 MOVEMENT 15
5.1.1 Movement control *15*
5.1.2 Collision detection *16*
5.1.3 Movement resources *16*
5.1.4 The IC Move behavior *17*
5.2 LOCATION FIRE 18
5.3 TARGETING THREATS 18

6. UNIT BEHAVIOR 19

6.1 SUPPRESSIVE FIRE 19
6.2 CLEAR ROOM 19

7. PVD ENHANCEMENTS 20

7.1 ICONS 20
7.2 MAP SCALE 20
7.3 MES BUILDING DISPLAY 20

8. MODSAF INTEGRATION 21

8.1 VERSION HISTORY 21
8.2 INTEGRATION DOCUMENTATION 22

9. SUPPORT ACTIVITIES 23

9.1 TEAM MEETINGS 23
9.2 EXPERIMENT SUPPORT 23
9.3 WEB SITE 23
9.4 TECHNICAL CONFERENCES 23

10. LESSONS LEARNED 23

11. REFERENCES 25

APPENDIX A. MES DEFINITION CONVENTIONS 27

GEOMETRY 27
TOPOLOGY 27
APERTURES 28

Executive Summary

Dismounted Infantry Semi-Automated Forces (DISAF) was developed as a component of the Dismounted Warrior Network (DWN) project. DISAF filled out a fireteam of four soldier trainees in Virtual Individual Combatant (VIC) platforms to a platoon of two squads and provided opposing force targets during DWN user exercises on a McKenna MOUT site database. The DWN exercise involved a team of four trainees assaulting and clearing a building.

The software development process started with the development by an SME of a Natural Language Combat Instruction Set (NLCIS) document, describing the required behaviors for the given task. Once each NLCIS was complete, it was turned over a DISAF software engineer who then transcribed the NLCIS into a corresponding Software Combat Instruction Set (SWCIS). Once the transcription was completed, it was reviewed by the originating SME and the assigned SW Engineer, and any discrepancies were reconciled. Validation was performed when the unit-tested software was integrated into the DISAF baseline. Other software process tasks performed included configuration management and the maintenance of on-line documentation and a problem reporting database.

The McKenna MOUT database was developed both as a visual database and a ModSAF CTDB. Operations in the DWN exercise were inside a building; thus the database had to contain interior details for one building. The CTDB was built by converting the visual, OpenFlight database to a ModSAF MES (building) data file using AutoCAD. In order to read the OpenFlight data and write out the information in ModSAF format, an AutoCAD plug-in was developed. The resulting data file was merged with a terrain skin database using ModSAF's recompile program.

Since the DWN exercise required that a hole be blown in the building wall. A repolygonization algorithm developed at ARL was used in all the DWN systems to transform the wall in response to a DIS detonation PDU at the wall. Additional code modified the DISAF MES data structures to reflect the change in topology.

New IC entities were created for DISAF. These entities use ModSAF conventions for the most part, but use components and primitive behaviors that have been rewritten specifically for humans rather than vehicles. Components upgraded included the lower body ("hull"), upper body ("turret"), eyes ("sensor"), and infantry weapon ("ballistic gun"). These component upgrades added the capability to change posture and change weapon state. The physical representation of IC entities was improved to allow the entity's dimensions to change dynamically as it changes posture.

New unit behaviors were added to DISAF to move the IC entities precisely (needed inside buildings). Behaviors available from the first phase of DISAF were modified to provide unit suppressive fire behavior and individual "location fire." The latter was used by missile launcher ICs to target the building wall in addition to supporting the suppressive fire behavior. A fireteam clear-room behavior was implemented to coordinate the movements of four soldiers as they enter a room.

The DISAF plan view display was enhanced to show icons representing IC entities in different postures and weapon readiness states. The display resolution was increased to allow users to see the small IC entities in small areas (rooms) at a reasonable size. Point placement resolution was also increased to allow users to place and designate routes for ICs very precisely.

The DISAF project culminated with the integration of the DISAF ERT into ModSAF for release as ModSAF 5.0 in early 1999.

Additional DISAF project activities included weekly integrated project team meetings, oversight of a web site for DWN documentation, support of engineering and user exercises at Ft. Benning, and presentation of papers on DISAF at technical conferences.

1. Introduction

1.1 Project Background

Dismounted Infantry Semi-Automated Forces (DISAF) is a component of the Dismounted Warrior Network (DWN) project. The major thrust of the overall Dismounted Warrior Network DWN effort is to develop a set of requirements for dismounted infantry (DI) simulation to support both the Training, Exercises, and Military Operations (TEMO) and the Advanced Concepts and Requirements/Research, Development, and Acquisition (ACR/RDA) domains. The most recent phase of DWN, Enhancements for Restricted Terrain (ERT), builds upon the lessons learned from the previous DWN effort and focuses on restricted terrain applications, specifically military operations in urban terrain (MOUT).

The DWN ERT experiments are intended to compare and contrast the ability of the key features of different Virtual Individual Combatant (VIC) technologies to support DI task performance in a virtual MOUT environment. The intent of comparing these different technologies over different tasks is to document the capabilities of each in order to be later matched against functional fidelity requirements flowing from the fidelity analysis portion of the original DWN effort. The result is the beginnings of a catalog that match existing technologies and capabilities against simulation requirements, and the identification of areas where future technology development is required.

The DISAF being developed by SAIC for the DWN effort have provided supporting BLUFOR to the VICs during the user exercises, as well as OPFOR snipers during these exercises. The DISAF was also the object of investigation during the experiments to assess its performance during both portions of the experiments (engineering and user).

The DISAF capabilities to support DWN primarily consist of creating new individual combatants (ICs) and IC-based units, and expanding ModSAF behaviors at the fireteam and squad levels relative to the scenarios used for DWN.

1.2 General Requirements

The DWN experiment requirements motivated many of the tasks in the DISAF project. The DWN ERT exercises take place on a database representing the McKenna MOUT (Military Operations in Urban Terrain) site at Ft. Benning, Ga. The exercise used the interior of one building (labeled “Building A”); the terrain database therefore modeled the interior of this building in detail. It was necessary to develop a DISAF terrain database that included a detailed, Multi-Elevation Structure (MES) representation of Building A.

The soldiers in DWN are displayed using an animated human figure that can assume various postures, gaits, and weapon positions; likewise, the communication protocol (DIS 2.0.4) supported these multiple configurations. Because the human trainees could observe DISAF entities closely, it was necessary for DISAF to carefully model the use of different body and weapon positions. Much of the DISAF work thus concentrated on “primitive” behaviors for individual DISAF combatants.

The DWN exercise involved a team of four trainees assaulting and clearing a building as part of a two-squad force. The second team in the trainees’ squad and the entire second squad were made up of DISAF entities. Both squads moved to the building and participated in clearing rooms. DISAF entities thus needed to be able to maneuver through an MES building and enter rooms as a team. DISAF behavior work focused on individual movement and team room-clearing. Additional behaviors modeled fire at a location, needed for suppressive fire and for breaching walls and doors.

The DWN exercise required special, ad-hoc dynamic terrain capabilities: At the beginning of the exercise scenario, a DISAF soldier uses a missile launcher to blow a hole in the side of the target building. Later, a soldier shoots out doors with a squad automatic weapon (SAW).

1.3 Statement of Work

Tasks to enhance DISAF were tracked as part of an overall list of tasks for DWN. Table 1 lists the DISAF tasks and indicates which parts of this report discuss those tasks.

WBS	Description	Addressed in this report section
1.1.2	Meeting Support	9.1
1.2.1	DI SAF Enhancements (NLCIS, SWCIS, V&V, primitive functions)	2, 4, 5, 6
1.2.5	DWN Web Site Operation	9.3
1.2.6.1	Dynamic Terrain	3.2
1.2.8.3	Graphical MES Editor, Bldg A, and DISAF PVD	3.1
1.2.8.4	ModSAF Baseline Integration	8
1.2.8.6	DWN ERT Capstone Study	NA
1.3	Application Support	9.2
1.4.1	Data (Problem Report) Analysis	2.7

Table 1. Table of DISAF DWN tasks and their location in this report.

Tasks marked "NA" are not applicable for this table because this report is due before those tasks are to be performed.

1.4 Deliverables

The deliverables consist of hardware, software and CDRL items.

1.4.1 Hardware and Software

Development hardware remains in Orlando (Two SGI Indys and one Maximum Impact). DISAF version 3.0 software was installed at the Land Warrior TestBed at Ft. Benning. SAIC has delivered DISAF version 4.0, ModSAF 4.0 with DISAF enhancements, to the ModSAF integration team for integration into ModSAF 5.0.

1.4.2 CDRLs

Three CDRL items were developed for this delivery order. They are tabulated in Table 2. The CDRL items will be posted to the DWN web site, which can be found at the following address: "http://www.stricom.army.mil/PRODUCTS/DWN."

CDRL #	Description	CM Number	Date
AB02	DISAF MOUT Enhancements Final Report	ADST-II-DWNERT-9800259	9/30/98
AB05	DISAF Support Summary Report	ADST-II-DWNERT-9800091	5/30/98
AB06	DWN ModSAF Baseline Documentation	ADST-II-DWNERT-9800261	9/11/98

Table 2 DISAF CDRL

2. Software Development Process

This section presents our approach for developing the DISAF software.

2.1 Overview

Figure 1 presents the approach for developing DISAF behaviors. The process starts with a Natural Language Combat Instruction Set (NLCIS) document, describing the required behaviors for the given task.

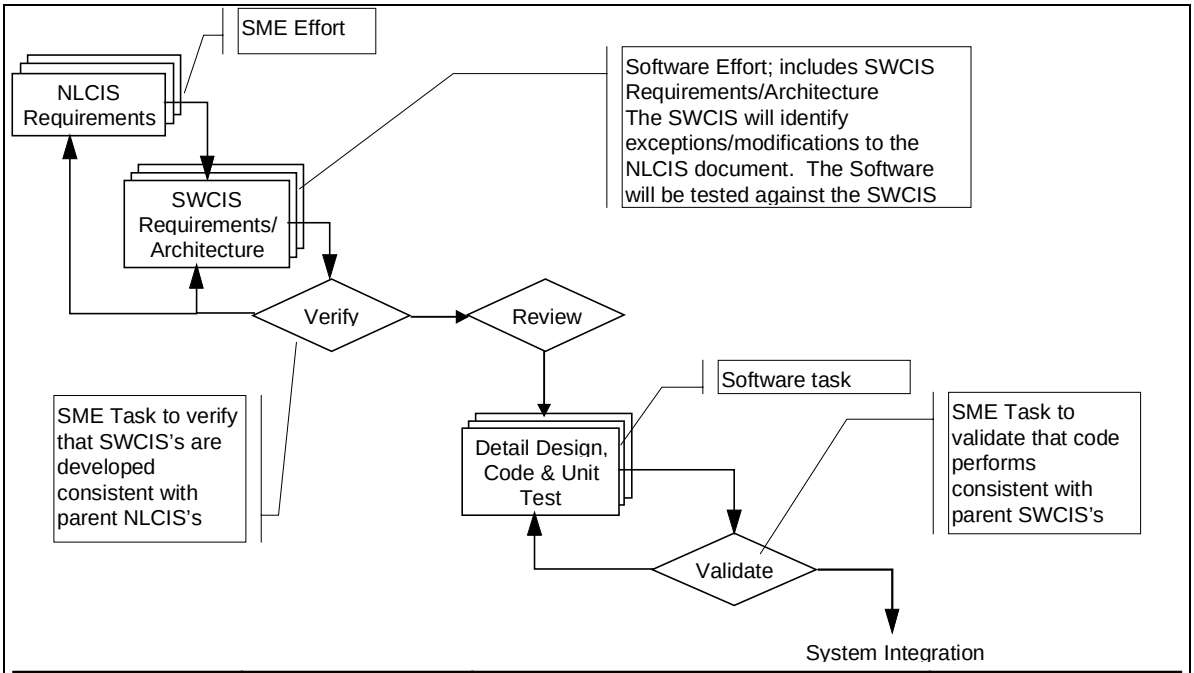

Figure 1 - DI SAF Software Development Approach

Once the NLCIS is complete, it is turned over to the DISAF software engineer responsible for implementation. The engineer then transcribes the NLCIS into a corresponding Software Combat Instruction Set (SWCIS). In general, this process consists of finalizing the requirements in the NLCIS until they are to a point that the SW design can begin. In addition, during this period, the SW engineer reviews the NLCIS requirements to determine if there are any that cannot be implemented. This might happen if the NLCIS requirement is not feasible given current technology (e.g. the requirement will require more computer cycles than the current hardware can support). Another reason for not including a requirement would be if it would require a design complexity that was beyond the scope of the present contract. In either case, such a requirement would be flagged for further review with the originator (SME) and with project management.

In some cases, the analysis by the software engineer will result in changes to the NLCIS. This would happen when the NLCIS is not complete or contains errors that were discovered by the software engineer.

For each SWCIS, once the requirements analysis is complete, it will be reviewed by the originating SME and the assigned SWE, and any remaining discrepancies will be reconciled.

2.2 Natural Language CIS

The purpose of natural language combat instruction sets (NLCIS) is to provide a sufficiently detailed description of real world combat behaviors that software engineers can successfully translate the description (tasks, conditions and standards) into requirements and code. NLCIS establish clear traceability to US Army doctrinal and tactical references, including ARTEPS, STPs, MTPs, and FMs. The development of the NLCIS is done by Army Subject Matter Experts, who are well-versed in tactical doctrine. A standard format for preparing CISs, used successfully during the development of the US Army's Close Combat Tactical Trainer (CCTT) and the British Army's Combined Arms Tactical Trainer (CATT) was used to write the NLCIS descriptions.

Table 3 lists the NLCISs developed for DISAF. These CISs were selected from a Task List developed for DWN when the exercise scenario was developed.

CIS #	Description
DI9004	Clear Room
DI0104	Enter Building and Clear Rooms (Squad)
TSK1004	Engage Targets With a M136 Launcher
DI0102	Perform overwatch/support by fire
DI9005	Clear Hallway
TSK1003	Engage Targets With M249 SAW
TSK2001	Engage Targets With M16A2 rifle
DI0101	Move tactically
TSK1001	Perform MOUT Movement Techniques
DI0105	Enter and Clear Building (Platoon)
TSK1005	Move as a Member of a Fire Team
TSK1006	Select an Overwatch Position
DI0106	React to contact
DI9002	React to Sniper Fire
DI0103	Execute assault
DI9003	React to Obstacles (rubble/walls)
TSK1002	Select MOUT Hasty Firing Position
TSK2002	Move Under Direct Fire
TSK2003	Select Temporary Fighting Positions
TSK1007	Control Movement of Fire Team
TSK1008	Control Maneuver of Squad

Table 3. DISAF Combat Instruction Sets

2.3 Software CIS

The SWCISs will constitute the top level of the software design, that is they will represent the "contract" between the SW developers, the SMEs, and the user community (as represented by Lockheed and STRICOM). The development of NLCISs into SWCIS might typically include

- converting symbolic requirements into numeric requirements (e.g., "near" ⇒ "less than 5 meters", etc.),
- identify timing and coordination constraints (e.g. event "A" has to occur before event "B")
- identifying spatial constraints (e.g. where an entity has to be at the beginning, end, and during a behavior)
- identifying coordination requirements (e.g. when one behavior controls another).

All of the NLCIS listed in Table 3 were converted to SWCIS. However, due to time constraints, only the top seven CISs in the table were implemented in DISAF.

2.4 Verification and Validation

Validation was performed when the unit-tested software was integrated into the current DISAF baseline. Both the SME responsible for the corresponding NLCIS behavioral description, and the software developers were involved. Typically a DISAF operator/developer would create scenarios and cause DISAF entities to perform tasks requested by the SME, who examined the operator tasks required and available and observed the resulting behavior on the DISAF plan view display. The SME requested tasks and behaviors that would demonstrate the activity described in the SWCIS.

2.5 Software Configuration Management

Software development involved dozens of ModSAF files and was performed concurrently by several developers. As a result, it was imperative that an adequate system for configuration management be employed, to maintain control over the evolving software configuration. The objective of a configuration management process is to track the implementation status on a file by file basis. As a result of using an adequate configuration management system, the developers can track the history of modifications to each file in ModSAF. The configuration management system that was chosen to use on the DISAF program was the Concurrent Version System, or CVS.

CVS is a CM tool that keeps a single copy of the master sources of a program and documentation. This copy is called the source “repository”; it contains all the information to permit extracting previous software releases at any time based on either a symbolic revision tag or a date in the past.

The “cvs edit <file>“ command allows a file to be modified, otherwise all files under CM are write protected.

After editing, sources are committed back to the repository with the “cvs commit <file>“ command. An editor is automatically started and the developer is required to document changes made to the file(s) before cvs will accept the file into the master.

The commands “cvs status” and “cvs update” allow the developer to check the status and receive updates from the archive. History on any file can be retrieved with the “cvs history” command to see the history of modifications along with the developer comments entered using the “cvs commit” command noted above.

In addition to log files maintained by CVS for each file, additional documentation is archived on a library or concept basis. This supports a forum for communicating issues about the file and even library level such as potential problem areas and new ideas. Text log files are also maintained with CVS.

2.6 On-line Documentation

During the development of DISAF, extensive documentation was developed and maintained on the same computer file system as the DISAF source code itself. Documents were written in HTML format, which allowed them to be accessed easily from any project computer using Web browser software. The HTML format allowed easy inclusion of screen captures to illustrate points and automatic hyperlinks to connect together related documents. A single top level page serves as the index into all of the documentation. The project maintained a variety of documents including open design questions, DISAF architecture documentation, code enhancements, notes from Technical Interchange Meetings, telephone and email contacts, short term task lists, etc.

2.7 Problem Tracking Report Database

Beginning with the validation of version 3.0 in July 1998, problems with DISAF reported by validating SMEs, users, and developers have been logged in problem tracking reports (PTRs). These PTRs are maintained in an HTML file and is available on-line. Periodically in IPT meetings the PTRs are prioritized based on how the problems affect approaching DISAF development milestones.

2.8 Software Metrics

The DISAF effort, including portions of open terrain behavior that were migrated from DISAF 1.1 to the ERT versions of DISAF, involved modifying or creating about 400 ModSAF files. These files contain over 267,000 lines of source code and data. Of these, approximately 28,000 are from the DWN Phase 1 effort, 36,000 are in new libraries for the ERT phase, and 203,000 are in files modified to support basic IC capabilities, IC definitions, MES operations, and other supporting functions.

3. Terrain Database Enhancements

3.1 Creating an MES for the McKenna MOUT TDB

3.1.1 Overview

This task was added to the project when it was discovered that it would not be possible to use an existing McKenna MOUT Site building that had previously been constructed as a MES model. It therefore became necessary to create another MES model (Building A) for the McKenna MOUT Site including the necessary development and visualization tools. A graphical editor was required to create Multiple Elevation Surfaces (MES) data structures from MultiGen Flight databases.

This section discusses the methods and tools used to develop an MES building for the CTDB database format used by ModSAF. The process was painstaking, involving numerous steps as shown in Figure 2. The steps are as follows:

1. The McKenna MOUT terrain skin, terrain features, and building A details were developed, debugged, and refined in OpenFlight format (visual database).

2. The terrain skin and features were converted to ModSAF CTDB format 7 with a Multigen software tool.

3. The OpenFlight building model was imported into AutoCAD using a custom-developed plug-in. These data were then used as a template to create the geometry of an MES building model. Naming conventions for the AutoCAD layers were used to establish the topology of the building.

4. An MES data file was written by AutoCAD, again using a custom plug-in.

5. The MES data file was edited by hand to add the transformation matrices needed to locate the building on the terrain, and to adjust aperture directions, etc.

6. The MES data file was merged into the CTDB of the terrain skin using the ModSAF recompile program.

This process is described in [Lockheed 1998].

3.1.2 MES Plug-in Development

Instead of creating an MES from scratch, the DWN MES plug-in can be used to import data from OpenFlight v14.2. The MES Plug-in was developed using the AutoCAD ObjectARX API and

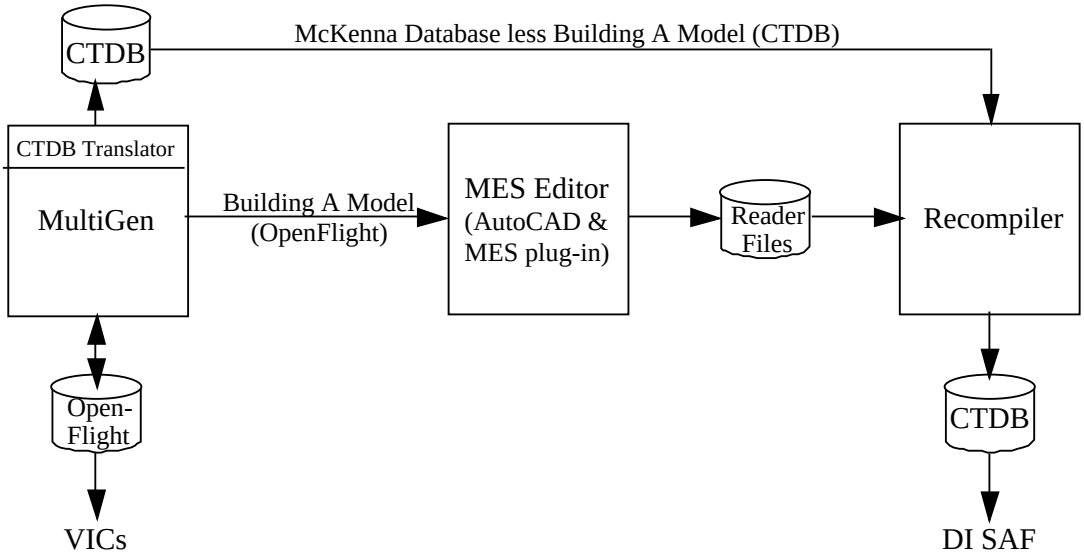

Figure 2 Process for creating TDB with MES building

development environment. Full documentation for the process of developing an ObjectARX application extension is provided in the ObjectARX documentation that is installed with the System Development Kit.

In the best case, the OpenFlight file will be structured in exactly the format of an MES building, and all that is required is to load the OpenFlight and then export to the .rdr file without having to do any geometry construction or modification in the interim. However, a visual model will typically NOT be constructed as an MES. More typically then, the OpenFlight geometry must be used as a template to locate the surfaces of each room and the MES must be constructed from scratch with the correct polygons.

In addition to creating the correct polygons, a user must separate the polygons in to apertures and enclosures. Each layer in AutoCAD corresponds to a single enclosure or aperture. In addition, certain naming conventions must be followed.

- **Enclosure layers** are always named starting with **Enc**.
- **Aperture layers** are always named starting either **Ad** to indicate a door aperture or **Aw** to indicate a window aperture. The next two components of the aperture layer name are separated by underscores '_' and indicate which two enclosures are connected by the aperture. The format is **`ad_<enclosure name>_<enclosure_name>`**
- A door aperture connecting two enclosures EncABC and EncDEF would be called Ad_EncABC_EncDEF. If multiple apertures connect the same two enclosures a trailing underscore and more text can be appended to the enclosure name to differentiate the apertures (i.e. Ad_EncABC_EncDEF_1, Ad_EncABC_EncDEF_2, etc.)

Once the MES is constructed, the plug-in can be used to write out the data as an MES .rdr file.

3.1.3 Editing the MES .rdr File

Once the .rdr file has been created, it must be edited to locate the building on the database. The .rdr file has two sections, a **template** section and a **volume** section. The template contains the geometry of the building with respect to some local coordinate system. The **volume** section tells the compiler where to instantiate the building and at what orientation. The same template can be instantiated multiple times by

having multiple volume sections in the .rdr file. The template section is created in the export process. The **VOLUME** section must be added by hand.

Other manual changes required in the .rdr file include the following:

- **Material Type -** The material types in the .rdr file must be set.
- **Wall_thickness –** This defaults to be 0.001 and 0.000. The value on the side of the aperture facing into the frame area should be set to the actual wall thickness.
- **Model name –** The name of the model always defaults to BuildingA. This may be changed for a different building.

3.2 Dynamic MES modification

Part of the DWN exercise involves having a fireteam enter a building through a hole blown by an AT8 missile launcher. In DWN this dynamic terrain problem was approached by distributing the calculation among all simulator nodes.

To create a hole in the local database of every simulator node in the exercise, each VIC and SAF computes the effect of a blast on the building independently—using the same algorithm and arriving at roughly the same result. An algorithm was available from the Army Research Laboratory (ARL) to repolygonize a wall to create a hole of specified size. For the VICs, this algorithm was sufficient to produce a modified visual database that showed a hole; for DISAF, the algorithm was modified to work with the different MES geometry and then extended to update the topological information stored in the MES. In particular, each hole created caused a new aperture to be created. This aperture had to have its own shape described with new triangles and an outline polygon, and it had to be connected to the room structures that were affected by the hole. Since the MES structure is built up from arrays of points, triangles, and other structures, inserting the new information required that the entire MES data structure be copied and the old one deleted.

Hole creation in DWN is triggered by DIS detonation PDUs. DISAF was modified to respond to detonation DIS PDUs by checking their proximity to building walls and triggering a hole-creation algorithm when the distance is below a given threshold. This threshold was set by DWN team members at technical interchange meetings. At these meetings it was also agreed that a “large building hit” result in the DIS PDU would be required to blow a hole; thus other detonation results do not trigger a proximity check in DISAF.

Another part of the DWN exercise requires that the SAW gunner from a fireteam shoot down a door to a room so that the fireteam may enter and clear the room. To implement this DISAF checks detonation PDUs to see if the munition path intersects a door. If so, the aperture attribute is set to “open.” The DWN project team agreed on the convention that only a detonation result of “medium building hit” could cause a door to be breached.

The M136 munition, fired by the AT-8 was the only weapon in the DWN that could cause a large building hit result. The M855 round causes a medium building hit, but only when fired by a SAW (not an M16A2). The DISAF IC weapon software library was extended to produce these detonation results for those weapons and munitions.

4. New IC Entity Definition

4.1 New entity types

DISAF implements a variant of a US Army mechanized infantry squad with an AT-8 for a missile launcher and with the grenadier replaced by a rifleman. Previously, ModSAF implemented a fireteam as a single simulation entity that was represented visually by a four-man icon. This project implements fireteams as individual soldiers. For friendly forces, a squad of 9 ICs is composed as follows:

	Role	Unit	Weapon	Munition
1	Squad Leader (Rifleman)	Squad	M16	M855
2	Fireteam Leader (Rifleman)	Fireteam A	M16	M855
3	Rifleman	Fireteam A	M16	M855
4	Rifleman	Fireteam A	M16	M855
5	SAW Gunner	Fireteam A	SAW	M855
6	Fireteam Leader (Rifleman)	Fireteam B	M16	M855
7	Rifleman	Fireteam B	M16	M855
8	AT8	Fireteam B	AT8 & M16	M136, M855
9	SAW Gunner	Fireteam B	SAW	M855

An OPFOR squad consist of a team of four ICs as follows:

	Role	Unit	Weapon	Munition
1	Squad Leader (Rifleman)	Squad	AK47	PS
2	Rifleman	Squad	AK47	PS
3	Rifleman	Squad	AK47	PS
4	Rifleman	Squad	AK47	PS

ModSAF's SIMNET origins show up directly in its internal type classification of "lifeforms." While tanks and other vehicles are discriminated by country, class, role, etc., life forms are all lumped together in one category. This situation creates problems in, for example, identifying threats to tanks; each missile-carrying life form must be listed individually rather than listing only a weapon class. For DISAF we have created the following new life form categories:

- Weapon Class
- Country
- Weapon
- Service
- Function

"Functions" are (nearly) permanent characteristics such as "combat" or "medical," rather than dynamic roles such as "squad leader" or "assistant gunner." Those roles are determined by the IC's position in its unit.

In designing these categories we considered eliminating the weapon as a type-discriminating feature, since it is easy to imagine a future requirement that would have entities picking up different weapons (for example, to keep automatic weapons in use). However, this approach was too much in conflict with DIS and the internal structure of ModSAF to be feasible in the time frame of the DISAF project.

4.2 Body Component Definition

ModSAF entities are defined in part by a set of physical components. These components define the simulation models and parameters for the hull or body, for guns, sensors, radios, etc. Our ModSAF IC entities are defined to have a "lowerbody," and "upperbody," "eyes," and an "icweapon." Figure 3 illustrates how these components form an IC.

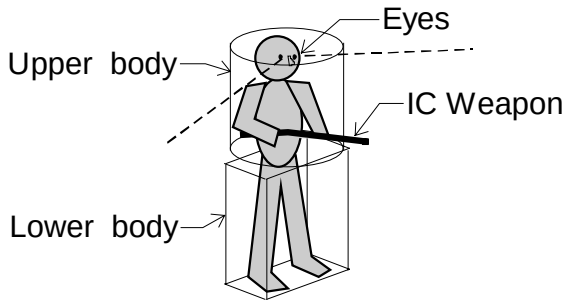

Figure 3. Components of an IC.

Classes of components in ModSAF are defined to other simulation modules in terms of interface functions. Each instantiation of the class should implement all of the interface functions; general behaviors can then be written without regard for which kind of instantiation they are controlling. However, in practice component functions and behaviors are often specific to certain types of entities such as aircraft or rotary-wing aircraft. DISAF too requires functional capabilities that are specific to ICs, such as changing posture or deploying a weapon. We have added this functionality without forcing artificial functions onto other entity types by defining new component classes which are derived from existing ones. Thus "LowerbodyDHull" is a class derived from the hull component class; it includes all of the hull functions and adds new IC-specific functions. "UpperbodyDTurret" and "ICWeaponDGun" similarly are derived classes. This approach crudely approximates language support for subclasses that is provided in languages such as C++.

4.3 Lower Body

The lower body component implements all movement functions of the IC. It is capable of moving forward, backward, or sideways. It maintains the posture state of the IC entity and changes posture between "standing," "kneeling," and "prone" by implementing new component functions.

The set-posture function is an important addition to the lower body model for two reasons. First, due to the animation of state changes in DWN, posture changes take a significant amount of time—e.g., 4.5 seconds for a standing to prone transition. DISAF ICs cannot change state instantly or there would be several seconds of error between the internal simulation of the IC and the remotely displayed rendering of the IC. However, the start of a posture change must immediately be reported over the network to other simulations so that they may begin their animations. Thus when a DISAF IC begins a posture change, the lower body simulation changes DISAF's network representation (in libentity) of the IC immediately but waits for some time before changing the internal state (in liblowerbody).

The second reason the lower body posture change function is important is that it takes away the ability of behaviors from changing the state of an IC directly in ModSAF's network representation. The lower body model can track health, damage, and other factors which can affect the time or ability of the IC to change postures; it is inappropriate for a behavior to shortcut this model.

Besides the posture states mentioned above, “crawling,” “crouching,” “walking,” and “running” are controlled by the lower body model. These “gaits” are not set by the set-posture function; rather, they are set automatically based on the posture and current speed of the IC. Crawling is prone plus a speed, crouching is kneeling plus a speed, and walking and running are standing plus a slow and fast speed, respectively. The speed thresholds at which the entity changes into these appearances are set by parameters.

Many capabilities of an IC are affected by its posture. The lower body model includes parameters to reduce the maximum movement speed when the IC isn’t standing, and to limit which postures are allowed when the IC has been injured. The lower body model will refuse to change postures if the IC’s speed or damage would not support the new posture.

4.4 Upper Body

The upper body component is analogous to the turret of a vehicle. It slews the weapon, and, at the present time, the eyes. We included this kind of component for two reasons: first, the ModSAF code upon which DISAF was developed assumes that weapons must be mounted on turrets, and we sought a good transition representation that would allow us to use existing behaviors; second, we wanted to provide the infrastructure that would, in the future, support the weapon aiming control capabilities available in the animation system used for DWN (DI-Guy, from Boston Dynamics Inc.—Koechling 1997).

The upper body component is designed to control the arms in general. Thus besides aiming the weapon, it deploys and stows it; it is also intended to deploy other objects (e.g. binoculars), throw grenades, and give arm signals. The upper body model is responsible for controlling the arm “resources” so that these actions cannot all be performed at once.

The new functions currently implemented for the upper body (beyond the existing turret functions) are

- Stow all
- Deploy component
- Drop component
- What is held?

As with the lower body’s posture change operation, the stow and deploy operations are animated on remote simulation nodes and take non-zero time to perform. These operations are implemented by changing the network representation immediately but changing the internal state only after a delay.

Unlike most tank turrets, the upper body “turret” does not rotate all of the way around. This physical limitation has implications for aiming the gun which ModSAF’s current gun model and search model ignore.

4.5 Eyes

The eyes are attached to the upper body. We have given them a fixed orientation and a field of view of 160 degrees. Within this field of view acuity and other characteristics are uniform.

4.5.1 Requirements

In DWN, SAF entities have close interaction with manned simulators--potentially moving in and out of view at ranges of less than 10 meters. We would like to approximate the capabilities of a human in this environment. In particular the simulated IC should, for nearby entities,

- detect entities that are visible only for a short time
- quickly detect entities that are in the periphery of the field of view
- direct its focus of attention—i.e., higher resolution vision—to detected objects in the peripheral part of the field of view.
- quickly identify targets near the center of the field of view.

For example, as the IC enters a room in a room-clearing task, it should be able to detect targets and engage them in less than a second.

4.5.2 Existing ModSAF model

The existing visual sensing model in ModSAF is based on a model from the Army's Night Vision and Electronic Systems Directorate (NVESD). This model is described by Lind (1995). It incorporates many factors affecting acquisition including apparent target size, atmospheric attenuation, and target contrast with background. It assumes that the observer is looking for targets by scanning a scene, and that it takes some time to scan the scene. The scanning pattern and scanning time as a function of field of view size are not modeled explicitly. Given a set of input factors listed above, the NVESD model produces a detection time. If the detection time is greater than the scene scan time, then detection does not occur at all. If detection does occur, the model determines the level of acquisition (e.g. just a detected entity, a class of vehicle, or exact type of entity).

In the ModSAF implementation of the NVESD model, an observer using direct vision is given a 10 degree (horizontal and vertical) field of view. Some types of observers—e.g. an M1 tank gunner—sweep their direction of gaze continuously with a vehicle component; others with "orientable" sensors—e.g. an M1 tank commander—can change their direction of gaze instantly to random directions. The sweep and redirection are controlled by a behavior that does not consider what bogeys or targets have been detected. For both types of sensors possible acquisitions are checked only every 3 - 5 seconds; in the case of orientable sensors, they are redirected before every check. Computed detection time for close or large targets is very small, on average, but the minimum time is currently limited to 1.0 seconds.

There are clearly several problems with the nominal ModSAF acquisition model, given the requirements outlined above. First, line of sight to undetected entities is sampled only once every 3 - 5 seconds, so entities not visible for that long could be missed completely. Second, there is no peripheral field of view; the 10 degree field of view effectively models only the narrow, high-resolution foveal vision with which targets would be identified. Third, search of the environment has no relation to objects or bogeys of interest. Fourth, acquisition of even the closest, most obvious target could take over 6 seconds—5 or more seconds for the next sample time, and then another 1 second for acquisition time.

4.5.3 DISAF modifications

In order to meet the requirements outlined above, it will be necessary to implement a revised visual sensing model in ModSAF. However, for the short time frame of DWN ERT, DISAF has only a modified version of the existing vision model. The modifications allow the DISAF ICs to meet the requirements reasonably well.

The first modification was to expand the field of view for ICs to 160 degrees. This gives them peripheral vision, although it is too good because it has the acuity of foveal vision. Second, the default scanning behavior was suppressed. This helps make up for the 160 degree foveal vision; in effect, the wide field of view is an aggregation of back-and-forth scanning. Turning off the scanning also prevents unrealistic use of the eyes to look to the side and to the front at the same time. ICs do not have a second set of eyes in their "hull" to drive with, unlike tanks. The third modification to the sensing model was to eliminate the 1 second minimum acquisition time. Finally, the 3 - 5 second sampling period was reduced to one half second. We have not yet determined the effect of this modification on system performance.

4.6 IC Weapon

4.6.1 IC Weapon capabilities

The IC weapon component is derived from the general ballistic gun model in ModSAF but is intended to simulate small arms that an IC manipulates with his hands. While the upper body performs the general deploy and stow operations on all such components, the components themselves determine more specific

actions with their interface functions. The most important function for IC weapons is the gun function already declared in ModSAF to set the position of a "launcher;" the IC weapon implementation uses this function to raise the weapon to firing position. As with the set-posture and deploy/stow functions above, this function sets the network appearance immediately but changes the internal state after a delay.

New IC weapon functions include the following:

- Get stow, deploy time
- Stow, deploy
- Set, get burst mode

The get-stow, deploy time functions provide weapon-specific delay times for the upper body component when it deploys or stows the weapon. The stow and deploy functions appear to be redundant with the stow and deploy functions of the upper body, but the IC weapon versions immediately toggle a flag in the weapon model. This flag, when indicating that the weapon is stowed, disables all weapon functions. The IC weapon stow/deploy functions are called by the upper body model when the internal state changes.

The burst mode for an IC weapon is typically "automatic" or "semi-automatic." Semi-automatic weapons fire only one round at a time—i.e., only one fire message, indicating one round, is broadcast from the simulation each time a behavior requests fire. The "quantity" argument in the gun's fire function is ignored. M16A2 rifles also have a "3-burst" mode which fires 3 rounds automatically. A bolt-action mode could also be supported by requiring an extra pause between rounds.

4.6.2 Weapon usage

Weapons fire for most entities in ModSAF is controlled through the IC Weapon model. In addition to the new features described above, we have made a number of changes to the nominal weapon firing process to accommodate small arms procedures and close quarters engagements.

One of the biggest features lacking from the existing ModSAF gun control model is the ability to rotate the body (hull) to aim the weapon. The existing gun controller directs the turret to rotate to the target and assumes that it can get there. If the turret slew limits prevent it from rotating far enough, the gun controller fails. We have added to the IC Weapon gun controller the ability to direct the body to rotate to aim the weapon.

ICs in DISAF are required, as part of one of their behaviors, to fire two rounds at a target. This is done with the fire selector on semi-automatic, so it represents two trigger pulls. ModSAF allows behaviors to specify how many rounds the gun is to fire, but the existing gun model implements this as an automatic fire burst of two rounds. The new IC Weapon model, as mentioned above, would not fire an automatic burst but would not fire two rounds, either. The two deliberate trigger pulls are properly modeled in a behavior rather than in the gun model; thus we have modified the nominal targeting behavior in ModSAF to accept a volley size parameter and to quickly repeat fire commands to the gun model until the complete volley has been fired. These rounds are fired regardless of whether the IC has lost sight of the entity or seen him change state to "destroyed."

4.7 Physical dimensions

4.7.1 Dynamic dimensions

The physical makeup of the IC shown in Figure 3 is modified when the IC takes different postures. For crouching and kneeling, the lower and upper bodies are shrunk by a percentage. The prone case, shown in Figure 4, is more complex. Here the upper body is moved beside the lower body. Both components are shrunk in height to match the standing depth. The overall depth is expanded to match the standing height.

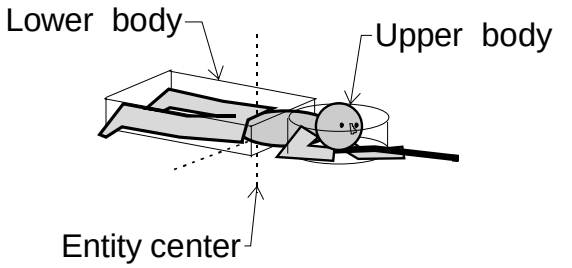

Figure 4. Prone IC configuration

In order to allow changes to entity dimensions during the simulation, a private copy of the physical database structure is created for each life form simulated by the local ModSAF or simulated remotely. Initially this structure contains a copy of the nominal physical characteristics; when the lower body model changes its posture state, however, modified dimensions are written into the entity's copy. When various libraries in DISAF request the entity's physical data, they are passed the private data instead of the static, global data.

The prone configuration has deeper ramifications for ModSAF than just a modification to dimensions. Some modules in DISAF (e.g., visual detection and hit determination) assume that the turret is always mounted on top of the hull. We have not as yet sought out all of the affected models, but we believe that any simulation errors caused by this assumption are outweighed by other inadequacies in the current models.

4.7.2 Dynamic sensor location

In addition to the physical dimensions, the location of the eyes changes as the entity changes posture. In the case of kneeling and crouching, the sensor location simply moves down as the entity gets "shorter." For the prone case, the eyes move forward and down. In all cases, the eyes are set a constant distance below the top of the entity. This modification is made in the sensor component model when the entity is a life form and the sensor name is "eyes."

5. Individual Behavior

5.1 Movement

The movements of ground vehicles in ModSAF is currently planned using a sophisticated temporal-spatial obstacle map and collision avoidance and reaction behaviors. In DISAF we are developing alternatives that are more appropriate to the scale of ICs rather than vehicles.

5.1.1 Movement control

We have found that the nominal movement control algorithm in ModSAF has a great deal of difficulty in the tight confines of an urban environment, either inside buildings or out in the streets. In this environment the movement algorithm is geometrically lacking because it sometimes produces paths that needlessly overshoot the waypoints, and sometimes fails altogether when there is a path available (see Figure 5). It is also not very good at maneuvering multiple entities in close quarters (such as tanks at a bridge). Tactically the algorithm is not adequate because it swings wide around obstacles (see figure below) rather than moving near them for cover. Practically the algorithm is difficult to use in new ways because it is very long and complex and has several parameters buried in it as constants—parameters that would need to change for an entity as different from a vehicle as an IC is.

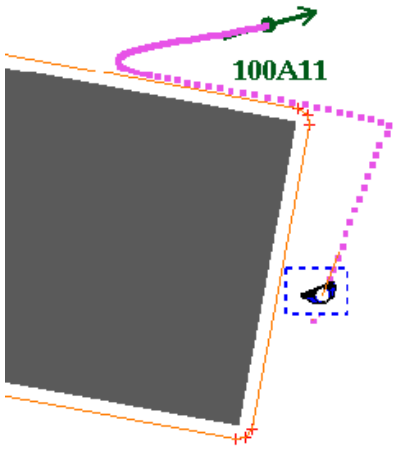

Figure 5. IC swinging wide around a corner under control of the nominal ModSAF algorithm

To achieve precise movement control in close quarters, we have developed an alternative movement control system that moves in a straight line between waypoints rather than trying to generate smooth, curving paths between them. This system is implemented in the Lower Body module described above. This movement controller does not plan routes or avoid obstacles; it is intended only to follow a precise path given to it by a movement behavior. The VICMove behavior, described below, was developed to provide such a path for IC entities.

5.1.2 Collision detection

Collisions in ModSAF are detected by a module that is independent of the movement model (lower body model). When a collision occurs it notifies the movement model and a collision response module. Collisions are expected to be rare events because vehicles are spaced far apart relative to their size and the nominal routes give wide clearance around obstacles (see above). A typical collision response involves bouncing back from the obstacle, stopping the entity, and possibly replanning a path. The response is not physically accurate; the entity is not stopped just at the point of contact with the obstacle. Furthermore, the response behavior runs much less frequently than the movement control and collision detection models so cannot respond to frequent collisions. For this reason multiple collisions with the same object are not reported if they occur within a refractory period of several seconds. Sometimes this results in ICs passing through walls on their second or third collision.

This nominal collision detection technique has previously been reported to be awkward for IC entities moving in formation (Howard et al 1995). A team of ICs moving in a building stresses the technique even more because they may frequently be within a body width of each other and actually brush against a wall as they move.

For ICs in DISAF the collision handling technique was modified in two ways. First, collision detection is more intimately connected to the movement model so that collisions are checked during every movement "tick" and the resulting IC position and velocity can be adjusted. Second, the collision response includes accurate adjustment of position to a point just prior to contact and velocity to a direction tangential to the obstacle. This allows ICs to have frequent contact with, for example, walls, without bouncing around excessively.

5.1.3 Movement resources

Unlike most vehicles, an IC does not have to face the direction it is moving. Furthermore, an IC can adopt different postures while it is moving. In some cases, it may be desirable to have different behaviors control different aspects of movement. For example, in section 4.6 we mentioned that the targeting

behavior had to control body orientation; the IC can shoot on the move if translation is controlled independently. We use this combination in the fireteam room-clearing behavior. Another example of multiple independent control is when ICs face a certain sector (with their bodies, not with their turrets as a tank would) while they move. We use this combination to face a trailing IC to the rear when a fireteam moves down a hallway.

To allow these independent motions, we extended DISAF's nominal "movement resource" to both a movement and an "orientation" resource. These resources are requested by behaviors and arbitrated by the Task Priority module. The moderated resources allow different behaviors to work together. For example, the behavior to engage targets controls the orientation of the IC if there is a target. The movement behavior VICMove requests orientation so that the IC can face his direction of travel normally, but defers to the engage target behavior if there is a target.

5.1.4 The IC Move behavior

Move is an entity or individual level behavior. The objective of this behavior is to provide precise movement, posture and weapon state control. Furthermore the objective is to provide human-like versus tank-like behavior generally used for all ground vehicles in ModSAF. This movement behavior is the underlying behavior for Clear Room behavior and is generally use inside buildings but can also be used on open terrain.

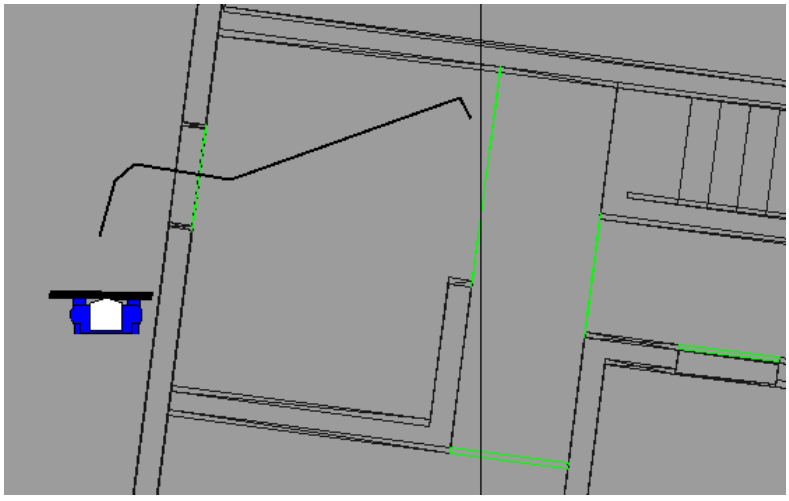

Figure 6 - Precise IC Movement Control

Figure 6 presents an example of movement control. An IC is prepared to enter a building through a door slightly wider then the width of the IC, travel close to a wall, spin 90° and stop, all without clipping the doorjamb, wall or any other object that might be in the room. Entity orientation is independent of direction of travel. It is desirable to have the ability to move sideways through the door jam or any other time during movement. In the case of engagement of enemy, the IC shall rotate and track the enemy independent of current movement.

This task is spawned from the Unit Operations Editor on an individual IC. It is also spawned from uclearroom. It provides public init, parser, etc. functions. Parameters that can be set from the GUI include:

- Route: either a point or a line. Note that if the route is a point, the IC entity will move directly toward it and not attempt to plan a route around obstacles.
- Start at beginning: A Boolean flag indicating that the entity should always move to the beginning of the route (line) rather than moving to the nearest point on the route to begin.

- Terminate at end: A Boolean flag indicating that the behavior should terminate when the end of the route has been reached. Otherwise, once the entity has reached the end of the route, it will move back to the beginning and repeat the move indefinitely.
- Stance: Allows the user to set the posture of the IC.
- Weapon state: Allows the user to set the state of the entity's weapon (stowed, deployed, raised).
- Speed: The speed at which the entity will travel.
- Maximum speed: The maximum speed that the entity can travel. Has no real effect currently.
- Angle mode: A Boolean flag indicating that the entity should always face the direction indicated by the angle parameter while it moves.
- Angle: The desired facing angle.

5.2 Location fire

The location fire task is an IC level behavior that is can be exercised directly from the GUI or utilized by unit level behaviors such as Suppressive Fire and Conduct Fire and Movement. It can be used by the SAW and AT8 to fire a single round at a building, door or window. Unit level behaviors use it to generate continuous fire over an area for a duration specified in minutes or seconds. It is important that the interface allow specification of the type of fire.

This task is spawned from the Unit Operations Editor on an individual IC. It is also spawned from usupprfire. It provides public init, parser, etc. functions. Parameters that can be set from the GUI include:

- Target location: this is a 3-dimensional location so that points off the ground can be specified.
- Elevation reference: the Z in the target location is relative to either the terrain elevation at the XY given in the location, or relative to the firing entity's elevation. The former is useful for most cases (it implements an "above ground level" reference), but the latter may be useful in a multi-elevation structure.
- Primary/secondary weapon: Refers to the primary or secondary weapon held by an IC.
- Stance: Allows IC posture to be set.
- Fire duration: Sets the duration of the location fire. "0" indicates one volley, however long it takes to fire it.

5.3 Targeting threats

In DISAF the existing ModSAF behavior for engaging threats, vtargeter, was extended in several ways. For ICs, vtargeter deploys the a weapon when there are no targets, just to make sure the IC is prepared. This only happens if no other task is trying to use the weapon; it is a default action. Also when no targets are visible, vtargeter makes sure that the weapon is not in the raised position.

Behaviors can set a firing "mode" via vtargeter by calling public interface functions. The first function sets the fire selector of the weapon, if one is available. The selector allows weapons such as the M16A2 to fire in a semi- or full automatic mode. (The M16A2, as determined by parameters to the IC Weapon module, can also fire in 3-round-burst mode.) The second function can cause vtargeter to fire more than one round at a target without checking, for the second and subsequent rounds, whether the target is still visible or whether the threat situation has changed. This mode of engagement is used in Room Clearing

behavior, per Combat Instruction Set, to make the IC shoot two rounds at each target. (This is in semi-automatic mode, not a two-round automatic burst.)

Finally, vtargeter controls the weapon position when engaging targets, making sure that the weapon of choice is deployed and then raised to the firing position before firing.

6. Unit Behavior

6.1 Suppressive fire

A IC, Fireteam or Squad may be required to provide suppressive fire on an enemy objective while another unit maneuvers towards the enemy location during an attack or assault on the objective. The unit's goal is to destroy or suppress the enemy and keep the enemy from firing on the assaulting force. Suppressive fires are distributed evenly over the objective. Continuous fire is maintained, even if no enemy weapons or positions are detected.

This task is spawned from the Unit Operations Editor on an IC fireteam or squad. It provides public init, parser, etc. functions.

The following data may be entered using the ModSAF GUI screen.

- Left_trp_tgt_area: Specifies the left target reference point. If the right_trp_tgt_area is not specified, then this point defines a single point target area.
- Right_trp_tgt_area: Specifies the right target reference point. If the same point is specified for left_trp_tgt_area or not specified, the target area is a single point as defined by the left_trp_tgt_area. Otherwise, the target area is a line between the left_trp_tgt_area and right_trp_tgt_area.
- Fire_duration_scale: Specifies the period that the Team should provide Suppressive Fire. Values of 0 indicate continuous firing.

The following parameters are configurable for the Suppressive Fire task at start-up time in the individual entities parameter files.

- Tick_period: specifies the length of a second (time) in milliseconds.
- Shooter_prep_time: Specifies the time in milliseconds that the Team needs to setup its gun and be able to fire.
- Battle_position_width: Specifies the width of the line that Team uses when it occupies its position. Determines how large an area will be searched to find a defilade position.
- Shooter_guise: Specifies the guise of the shooting vehicle in the Team. A value of 0 indicates all DIs in the Team will be shooting.

6.2 Clear room

Clear Room is a Fireteam level behavior derived from the Software Combat Instruction Set DI 9004. It is designed specifically for an Army four-person fireteam. The purpose is to enter a room and engage enemy with weapons fire. Each fireteam member enters the room in a specific order and moves to a specific location, visually searching specific sectors (see Figure 7). The behavior ends when ICs have reached their final positions in the room to be cleared. Room clearing is performed by control of posture (stance), weapon state, rules of engagement and movement. For the duration of Clear Room, ROE is set to "volley" and "free", stance is set to kneeling (crouched when moving) and weapons are raised.

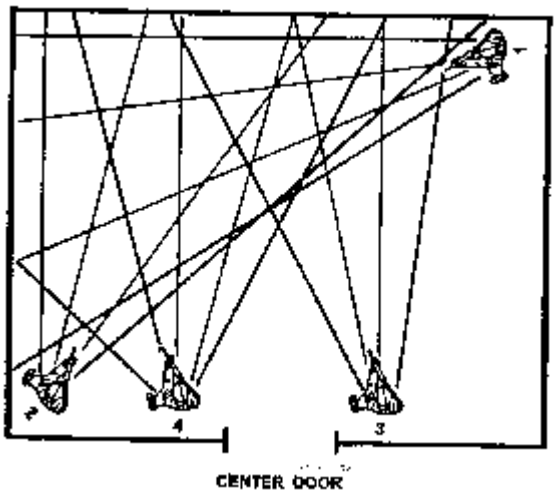

Figure 7 - Clear Room, Center Door. From [Dept. of the Army 1993]

Inputs to the Clear Room editor can be split into two groups, 1) movement to Ready Positions and 2) movement into the room. The first group is the column headed by #1 M16 Ready Position. These four positions define starting positions for each member of the fire team. A speed input parameter defines how fast ICs should travel to the Ready Positions. During this movement, ROE is set to free, weapons are raised and stance is set to crouch (kneeling with movement). The second group starts with Route through door and continues with the column headed by #1 M16 Final Position. At 1/2 second intervals and in order (#1, #2, #3, and #4), each IC first follows the Route through door then a route to a final position. Route through door is optional.

7. PVD Enhancements

7.1 Icons

In order to see more precisely what the DISAF ICs are doing, and to see how they are moving in the confined spaces of building interiors, the IC icons on the ModSAF Plan View Display (PVD) were replaced. The stick figure that was used to represent ICs in the current version of ModSAF was replaced with a bird's eye picture to be consistent with other entities and to show the true space occupied by the IC. Different appearances were designed for standing, crouching, kneeling, and prone postures; in each posture, the stowed, deployed and in-firing-position states are different. Figure 8 shows the new icons.

7.2 Map scale

In addition to adding new icons, the resolution of the PVD was increased. The minimum display scale is now 1:25, and it is possible to place points on the display to resolutions less than 1 meter. These changes required updating the internal representation (type and width) of graphic coordinates in ModSAF. Both of these new capabilities are necessary for a DISAF operator when the entire area of engagement is only 20 meters or so on a side!

7.3 MES building display

The ModSAF PVD has been extended to show the interior of MES buildings. This display is necessary for placing ICs inside buildings, placing routes and other objects inside buildings, and understanding what

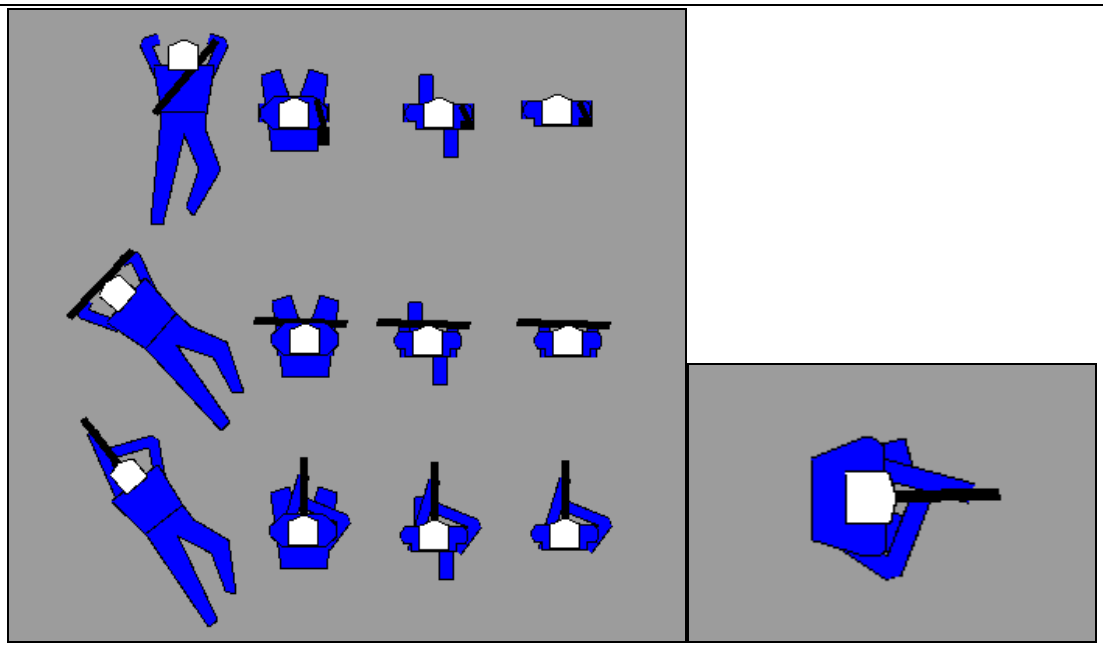

Figure 8. Left: new ModSAF IC icons for (left to right) prone, sitting, kneeling and standing, and (top to bottom) stowed, deployed, in-firing-position. Right: Crouching posture with weapon in firing position, used for the "quick kill" posture required by the room clearing behavior.

entities inside buildings are doing. The display not only shows interior walls, but marks apertures indicating whether they are doors or windows and whether they are open or shut. Figure 9 shows the appearance of the display as of the time of writing.

8. ModSAF Integration

DISAF ERT functionality was integrated with ModSAF 4.0 in August and September 1998. Thus DISAF will be available to the community when ModSAF 5.0 is released in early 1999.

8.1 Version History

Date	Event	DISAF version	Based on ModSAF version
8/97	End of phase 1 DISAF development (open terrain behaviors)	1.0	MC SAF
9/97	Partial integration of DISAF modifications into ModSAF	2.0	STOW 2.2.4
3/98	ERT enhancements to support April 98 demo	2.2	
7/98	ERT enhancements to support user experiments	3.0	
9/98	Pre-integration into ModSAF 4.0	4.0	4.0
10/98	ModSAF 4.0 with DISAF and all other integrations	5.0	5.0 pre-release

Figure 9. ModSAF PVD showing MES interior

8.2 Integration Documentation

As part of the ModSAF integration task, five documents were written:

- **Software Requirements Specification** [SAIC 1998a]. The Software Requirements Specification (SRS) describes the requirements of ModSAF capabilities and the methods to be used to ensure that each requirement has been met. Furthermore, the SRS is used as the basis for design and qualification testing of new ModSAF capabilities.

- **Conceptual Model Document** [SAIC 1998b]. A conceptual model describes the domain (real-world) aspects which pertain to new capabilities desired for ModSAF, whereas a Software Requirements Specification describes the scope and fidelity of only the relevant aspects of the domain required for the simulation software. Domain information is usually obtained during "knowledge acquisition" and "knowledge engineering" activities by Subject Matter Experts.

- **Software Design Description** [SAIC 1998c]. The Software Design Description (SDD) describes the design of ModSAF capabilities. It describes the capability-wide design decisions, the architectural design of the capabilities, and the detailed design needed to implement the software. The SDD is used as the basis for implementing the software. It provides the acquirer visibility into the design and provides information needed for software support.

- **Software Test Description** [SAIC 1998d]. The Software Test Description (STD) describes the test preparations, test cases, and test procedures to be used to perform testing of ModSAF SRS requirements. The STD is used as the basis for proving that specific requirements have been satisfied by the corresponding implementation.
- **DISAF Users' Manual.** While not officially part of required ModSAF integration documentation, the Users' Manual is nevertheless an important part of DISAF documentation. The Manual explains with text, figures and screen captures how the user can use the behaviors in DISAF and what to expect from them.

9. Support Activities

9.1 Team Meetings

Throughout the DISAF project, Integrated Project Team (IPT) meetings were held including representatives from STRICOM, Lockheed Martin, and SAIC. These meetings were used to discuss and prioritize the many technical issues encountered on the project. In addition, the DISAF team participated in two Technical Interchange Meetings (and associated "pre TIMs") to discuss DWN-wide technical issues with all contractors.

9.2 Experiment Support

SAIC delivered DISAF, Dismounted Infantry Semi Automated Forces, to Ft. Benning in July to support engineering and user experiments in the Dismounted Warrior Network (DWN) project. During these experiments, teams of soldiers from Ft. Benning on simulator platforms assaulted and cleared a virtual building with the help of three fireteams of SAF soldiers. A DISAF soldier also breached a building wall with a missile to provide the entry point.

After the experiments were completed, a media day was held for local and national news organizations. DWN was featured on CNN Headline News, ABC News, and Columbus, GA news programs.

9.3 Web Site

Documents generated as part of the DWN program were made publicly available via a site on the World Wide Web. This site was maintained by Resource Consultants, Inc. (RCI) under subcontract to SAIC from the beginning of the ERT phase (9/97) until the end of January, 1998. At that time the content of the site was made available and maintained by STRICOM as part of their site. The computer used by RCI to provide the site was later transferred back to STRICOM.

9.4 Technical Conferences

Over the year of the DISAF ERT project, various members of the project team attended the Spring Simulation Interoperability Workshop (SIW) in March 1998, the Computer Generated Forces and Behavioral Representation Conference in May 1998, and the Fall SIW in September 1998. Papers were presented at all three ([Ourston 1998], [Reece 1998a], [Reece 1998b]).

10. Lessons learned

- MES Creation. The MES creation steps are described in Section 3.1.1. While the process is documented and repeatable, it is tedious and error prone. Especially difficult is step 3, which requires an AutoCAD operator to reconstruct the building by hand using the OpenFlight geometry as a guide.

We encountered several problems in our MES (which were corrected) due to small errors in the placement of triangle vertices. The operator must also use care in naming layers so that the correct topology is generated. Any errors in these steps can result in an illegal (in terms of the CTDB definition) MES, a mal-formed MES that will cause errors during elevation lookups etc., or even just an MES whose geometry does not correspond to the visual model. It would be preferable to automate the OpenFlight to MES process.

It is also worth noting that during the original creation of Building A, the MES creation steps were performed by four different organizations in three different cities. When problems were discovered in the final database, they had to be traced back through the creation steps from organization to organization. A fix early in the process then had to be progagated forward again. Each iteration of this process took a long time.

- The footprint field of an MES enclosure structure is not useful. If the room is concave, then the footprint (a convex hull in the xy plane) may overlap other enclosures.

- Doors are not represented in the MES definition. Also, there is no easy way to model furniture inside an MES. These problems should be addressed to make MOUT simulations with DISAF more realistic. As it its, it is not possible to replicate the actual McKenna MOUT site within DISAF.

- MES structures are dynamic: apertures can have their characteristics changed (mobility, opacity), and in DISAF doors and walls can be breached. Unfortunately there is no way to save or restore the state of an MES building in DISAF. There is no way for a user to open or close doors from the GUI to set up a scenario. Neither is there a way for entities under behavior control to open doors or windows (except for shooting them out permanently). We found that in many cases it was inconvenient to work with the MES in a scenario because of this lack of control over the dynamic aspects of the MES.

- The MES-specific routines in terrain library in DISAF were based upon some bad assumptions about MES's, resulting in bugs and crashes. We believe this was partly due to the fact that the developers of the MES code did not have a good MES to test with and did not have entities that they could move easily around buildings. Nevertheless, it was a problem for this project. We anticipate that additional DISAF development in the MOUT environment will uncover additional problems as the MES interface functions are exercised more.

- There are numerous aspects of buildings that may legally be encoded in MESs in several different ways. For example, apertures (notional boundaries between rooms) may be encoded anywhere within the door frame. The choices made could conceivably have an impact on MES terrain reasoning algorithms; for example, a search for cover (the door frame area) within a room. Some conventions for design were developed in the DISAF project (see Appendix A). Additional conventions will be needed as more behaviors are developed for MESs.

- Although not originally part of the DISAF-ERT project plans, the numerous PVD enhancements that wereimplemented (see Section 7) turned out to be crucial for development, debugging, scenario development, and user control. Further enhancements to the PVD are necessary, as it is difficult to distinguish between entities on different levels and there is no way to display different buildings independently.

- The DWN simulators used the DIS 2.0.4 protocol to represent ICs on the network. ICs thus were represented in terms of a posture or gait (standing, walking, running, crouching, kneeling, prone, or crawling), a weapon state (weapon 1 or 2 stowed, deployed, or raised) and a velocity. The human figure animation system, DI-Guy, displayed a moving figure corresponding to the DIS state. Since not all states were available via DI-Guy, some interpretation had to be made. For example, there is no kneeling-with-velocity animation, so kneeling + velocity is interpreted as crouching + velocity. In spite of several discussions at TIMs and via email, the DWN systems did not all have uniform interpretations during the experiments. Further effort should be made to standardize IC states and animations.

- The human figure animations used by the VICs showed ICs changing state realistically—over the course of several seconds. DIS was designed with the assumption that state changes would be instantaneous. As a result, simulations using the figure animations can lose coherence during the transitions. DISAF delayed its internal state change to match the remote animations, as discussed in Section 4.3. However, this delay would be inappropriate for interaction with another DISAF system because the remote DISAF would change the IC state instantly. What is required is an extension to the network protocol—perhaps via a new HLA FOM—to allow simulations to indicate when transitions were starting and how fast they would take place, or when they ended.

- The DISAF ERT project team originally envisioned implementing numerous unit MOUT behaviors (see the CIS list in Table 3). However, the majority of the effort went into developing "infrastructure" and "primitive behaviors." This work included modification and additions to the terrain functions, the console command set, the internal entity definitions, the weapons model, the movement control model, the plan view display, the user interface editor, and other fundamental areas of DISAF. A look at the number of lines of code involved in modification vs. new behavior code (see Section 2.8) shows that approximately six times more code was involved in DISAF modification than in new code development. The lesson to be learned here is that developing a capability for a new type of entity like and IC requires a great deal of effort for modeling the physical or primitive aspects. We believe that future enhancements of DISAF will require even more work in these aspects along with behavior additions.

11. References

Department of the Army (1993) *An Infantryman's Guide To Combat in Built-Up Areas.* Field Manual No. 90-10-1.

Howard, M., Hoff, B., and Tseng, D. (1995). "Individual Combatant Development in ModSAF." In the *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation.* University of Central Florida.

Koechling, J. (1997) "Live Reckoning for Simulated Dismounted Infantry using DI-Guy(tm)". In *Proceedings of the 1997 Fall Simulation Interoperability Workshop*, SISO.

Lind, J. (1995). *Target Acquisition Models for* Janus (A). NAWCWPNS TM 7811, Naval Air Warfare Center Weapons Division, China Lake, CA.

Lockheed Martin Information Systems (1998). *MES Editor and DISAF Support: Summary Report.* CDRL AB05 of Delivery Order #55. Lockheed Martin Information Systems Company, Orlando, FL.

Ourston, D., D. Reece, and P. Dumanoir (1998). "Issues Involved With Integrating Live and Artificial Virtual Individual Combatants." In *Proceedings of the 1998 Spring Simulation Interoperability Workshop*. IEEE.

Reece, D., D. Ourston, M. Kraus, and I. Carbia. (1998a) "Updating ModSAF for Individual Combatants: the DISAF Program". In *Proceedings of the 7th Conference on Computer Generated Forces and Behavior Representation*. University of Central Florida.

Reece, D. and P. Dumanoir (1998b) "Conventions for Representing Humans in a DIS Exercise: the DWN Experience". In *Proceedings of the 1998 Fall Simulation Interoperability Workshop.* IEEE.

SAIC (1998a) *DWN ModSAF Baseline Documentation, Part 1: Software Requirements Specification.* CDRL AB06 of Delivery Order #55. Lockheed Martin Information Systems Company, Orlando, FL.

SAIC (1998b) *DWN ModSAF Baseline Documentation, Part 2: Conceptual Model Document*. CDRL AB06 of Delivery Order #55. Lockheed Martin Information Systems Company, Orlando, FL.

SAIC (1998c) *DWN ModSAF Baseline Documentation, Part 3: Software Design Description*. CDRL AB06 of Delivery Order #55. Lockheed Martin Information Systems Company, Orlando, FL.

SAIC (1998d) *DWN ModSAF Baseline Documentation, Part 4: Software Test Description*. CDRL AB06 of Delivery Order #55. Lockheed Martin Information Systems Company, Orlando, FL.

Stanzione, T. *et al*. (1996). "Multiple Elevation Structures in the Improved Computer Generated Forces Terrain Database" In the *6th Conference on Computer Generated Forces and Behavioral Representation*. Institute for Simulation and Training.

Appendix A.
MES definition conventions

Although DISAF has not yet been extended to incorporate many new terrain reasoning algorithms dealing with MES's, several ambiguities in the MES definition did become apparent when working with the existing algorithms. This section documents several aspects of MES building that we feel should be standardized so that future algorithms can depend on a standard meaning for the definition.

An MES is a collection of geometry and topology information which describes both the physical geometry of a multi-elevation structure (typically a building) and the connectivity of different parts of the structure (typically rooms). The geometry of the structure is broken down into distinct volumes called *enclosures.* Typically, an enclosure is comprised of an individual room within a building. The topology of the structure is represented by the connections between enclosures. These connections are called *apertures.* Many details of MES representation are described in [Stanzione 1996].

Geometry

- Enclosure triangles must not intersect, be shared, or be overlayed (coplanar) with triangles from any other enclosures. The volumes for enclosures should be distinct, except for enclosure 0, which contains all the other enclosures.
- Enclosure triangles are always defined such that their front face is defined by a counter-clockwise winding of the vertices of the face. The front face of enclosure triangles always points **inward** to the enclosure.
- There should never be holes or gaps in an enclosure geometry, except for the apertures which connect to another enclosure
- Any edges of enclosure faces must share vertices at the endpoints of the edges – there can be no T-vertices.
- Enclosures should normally be constructed so that they include the parts of door and window frames up to the physical door or window. I.e. if the actual door is in the middle of the door frame between two rooms, then the enclosure for each room should include polygons defining half of the door frame. If the door is actually one side of the door frame, then one enclosure should contain door frame polygons and the other shouldn't. If the aperture is a doorway or opening without a physical door or window, then the door frame should be placed entirely in one enclosure. This placement aids in reasoning about how such a frame may be used for movement and cover.
- The footprint of the enclosure is the convex hull in the XY plane of the enclosure. Note-- since this is defined in the reader file, before translation to world coordinates, there is no guarantee that the footprint in world coordinates (in the libctdb data structure) will be in the XY plane.
- A special case is the first enclosure, enclosure 0 (zero), which is the exterior of the MES. Enclosure 0 only contains apertures which connect it to enclosures on the inside of the structure. Every MES must have an enclosure 0. The triangles for enclosure 0 should be oriented such that their front faces point **outwards** from the MES.
- Apertures are defined as an N-sided polygon outline at the juncture of two enclosures. The N-sided polygon should be convex. The aperture structure also contains polygons which comprise the triangulation of the outline.

Topology

- An aperture always connects exactly two enclosures together.
- All enclosures should normally connect to at least one other enclosure via an aperture.

Apertures

Apertures have the following attributes:

Attribute	Description
Material	Material type of the aperture (glass, etc.). There is no given standard enumeration for type.
Open	Boolean 0/1 indicating if aperture is penetrable (can be moved through)
Visible	Boolean 0/1 indicating if aperture is transparent
Wall thickness (Two—one for each connected enclosure)	(2) entries – thickness of wall on either side of the aperture. Note that the first thickness value is in the direction of the aperture, i.e. the normal vector of the aperture outline polygon.
Step to enclosure (Two—one for each connected enclosure)	Boolean 0/1 indicating if there is any step down from the aperture to the enclosure (i.e., usually 0 for doorways).

Apertures should normally be created right at the edge of where one enclosure butts up against another, as opposed to in the center of a doorway or window. In this model, that would mean that one of the wall_thickness parameters is always 0, and the other is the thickness of the door or window frame. See Figure 10.

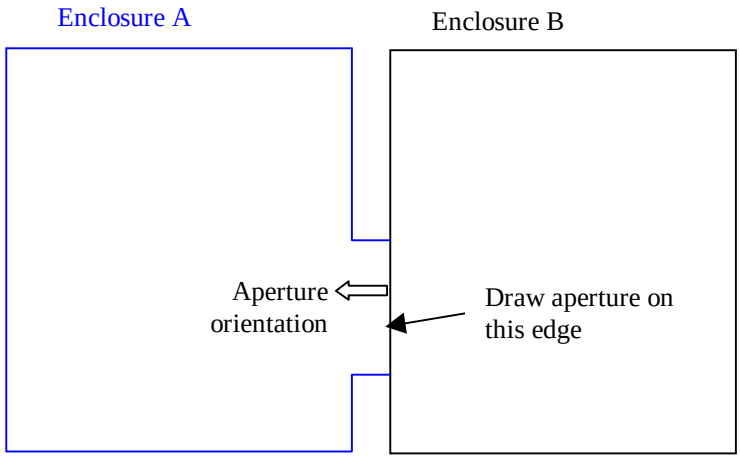

Figure 10. Aperture orientation convention